Image and Vision Computing 28 (2010) 1130–1145

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Online learning of task-driven object-based visual attention control

Ali Borji [a,b,*], Majid Nili Ahmadabadi [a,c], Babak Nadjar Araabi [a,c], Mandana Hamidi [d]

[a] School of Cognitive Sciences, Institute for Research in Fundamental Sciences, Niavaran Bldg., P.O. Box 19395-5746, Tehran, Iran
[b] Dept. of Computer Science III, University of Bonn, Bonn, Germany
[c] Control and Intelligent Processing Centre of Excellence, Dept. of Electrical and Computer Eng., University of Tehran, Tehran, Iran
[d] Italian Institute of Technology (IIT), Via Morego 30, 16163, Genova, Italy

ARTICLE INFO

Article history:
Received 5 October 2008
Received in revised form 4 August 2009
Accepted 9 October 2009

Keywords:
Task-driven attention
Object-based attention
Top-down attention
Saliency-based model
Reinforcement learning
State space discretization

ABSTRACT

We propose a biologically-motivated computational model for learning task-driven and object-based visual attention control in interactive environments. In this model, top-down attention is learned interactively and is used to search for a desired object in the scene through biasing the bottom-up attention in order to form a need-based and object-driven state representation of the environment. Our model consists of three layers. First, in the early visual processing layer, most salient location of a scene is derived using the biased saliency-based bottom-up model of visual attention. Then a cognitive component in the higher visual processing layer performs an application specific operation like object recognition at the focus of attention. From this information, a state is derived in the decision making and learning layer. Top-down attention is learned by the U-TREE algorithm which successively grows an object-based binary tree. Internal nodes in this tree check the existence of a specific object in the scene by biasing the early vision and the object recognition parts. Its leaves point to states in the action value table. Motor actions are associated with the leaves. After performing a motor action, the agent receives a reinforcement signal from the critic. This signal is alternately used for modifying the tree or updating the action selection policy. The proposed model is evaluated on visual navigation tasks, where obtained results lend support to the applicability and usefulness of the developed method for robotics.

1. Introduction

Both biological and machine vision systems have to process enormous amount of visual information they receive at any given time. Attentional selection provides an efficient solution to this information overload problem by proposing a small set of scene regions to higher level and more computationally intensive processes; like scene interpretation, object recognition, decision making, etc. In this regard visual attention acts as a front-end to a more complex vision system. Instead of processing all incoming visual information in parallel, the brain has evolved a serial strategy which explains its near real time performance in visual interactive environments.

Visual attention selects and gates visual information based on the saliency in the image itself (bottom-up) [1,2] and on the prior knowledge about the scene (top-down) [3,4]. While bottom-up attention is solely determined by the image-based low-level cues – such as luminance and color contrasts, edge orientation and motion – top-down attention on the other hand is influenced by task demands, prior knowledge of the target and the scene, emotions, expectations, etc. Bottom-up component of the visual attention is mainly examined by the early visual areas of the brain like LGN and V1 [6]. Top-down attentional signals are largely derived from a network of areas in parietal and frontal cortex [5]. Some of the involved areas include the superior parietal lobule (SPL), the frontal eye fields (FEF), the supplementary eye field (SEF) and the lateral prefrontal cortex in the region of the middle frontal gyrus (MFG). In daily life, these two mechanisms interact to control our attentional behaviors [3,7]. Besides acting in spatial domain by selecting spatial locations [8], visual attention can also be directed to particular features such as color, orientation and direction of motion [9]. It is also believed that attention selects objects rather than spatial locations [10,11].

Like humans and primates, artificial creatures (e.g. robots) are limited in terms of allocation of their resources to huge sensory and perceptual information. That is mainly because of the serial processing mechanisms used in the design of such creatures which allows processing of only a small amount of incoming sensory information. Since they are usually supposed to guarantee a short response time, attention is an efficient solution in robotics as in biological systems. In order to gain the maximum cumulative reward in the minimum time, agents should be able to perform perceptual and physical actions simultaneously. These perceptual

* Corresponding author. Tel.: +98 21 22294035; fax: +98 21 22280352.
E-mail addresses: borji@iai.uni-bonn.de, borji@ipm.ir (A. Borji), mnili@ut.ac.ir (M.N. Ahmadabadi), araabi@ut.ac.ir (B.N. Araabi), Mandana.hamidi@iit.it (M. Hamidi).

0262-8856/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2009.10.006

actions are available in several forms like where and what to look in the visual modality. However, the main concern is how to select the relevant information, since relevancy depends on the tasks and the goals. In this study, we consider task relevancy of visual information and aim to extract objects which help the agent to discover its state faster for decision making.

It is important that a solution for learning task-based visual attention control to take into account other relevant and dependent cognitive processes like learning, decision making, action selection, etc. Some evidences in this regard exist in both biology and engineering. It has been previously shown that attention and eye movements are context-based and task-dependent [12]. Previous experiences also influence attention behaviors which indicate that attentional control mechanisms can be learned [13]. Some neuropsychological evidences suggest that human beings learn to extract useful information from visual scenes in an interactive fashion without the aid of any external supervisor [14]. In [15], it has been shown that attention is also affected by decision behaviors. These findings are in accordance with a new and pragmatic view in Artificial Intelligence (AI) known as embodied and situated intelligence [16]. It states that intellectual behaviors, representations, decisions, etc. are the product of interactions among brain, body and environment. In [52], authors have provided their views for architecture of situated vision systems, how to tackle the design and analysis of perceptual systems and promising future directions. In particular they have focused on inspiring from complex vision systems, like human vision, to build synthetic vision systems and integrating them with action and learning modules. There are also other supporting evidences in psychology claiming that human mind and intelligence have been formed interactively through an evolutionary process [17]. Instead of attempting to segment, identify, represent and maintain detailed memory of all objects in a scene, there are evidences that claim our brain may adopt a need-based approach [18], where only desired objects are quickly detected in a scene, identified and represented. Considering the above evidences, in this work, we introduce a model to consider the influences of task, action, learning and decision making to control top-down visual attention of an agent.

In many real-world situations, the environment is unfamiliar or not clearly defined. Moreover, required information and the optimal responses are not known at the design time. Therefore, fixed and predefined design of attention control strategies in such situations is less useful. Some complicated behaviors of humans like reading, writing, driving, etc. which need complex physical actions and attentions witness that such behaviors have been developed based on humans interaction with the surrounding world. Thus, interactive and semi-supervised approaches, e.g. Reinforcement Learning (RL) [19], seem to be the most suitable techniques for learning top-down visual attention control and action selection strategies. Such learning mechanisms have the benefit of adapting the agent to dynamic, complicated and non-deterministic environments. In RL, agents learn action-values in each state by receiving a reinforcement signal from the critic. Another characteristic of RL methods is their ability of online learning which is required for interacting with stochastic and slowly changing environments. There are mathematical convergence proofs for these methods and they are biologically plausible [20].

Our proposed top-down visual attention model is built upon a sound and widely used bottom-up visual attention model proposed in [5,21]. This model is based on the idea of saliency map, an explicit 2D topographical map that encodes stimulus conspicuity or saliency at every scene location [22]. Bottom-up model in its original form is solely data-driven and simply selects some spatial locations without using any feedback mechanism or top-down gains. Some researchers have tried to add top-down capabilities to this basic model [23–25], for instance by biasing it toward selecting specific objects. While such models are interesting, they have been partially successful to handle a limited category of tasks. Modeling top-down task-based influences on visual attention is difficult, since a general definition for a task does not yet exist. In this study, RL is used by the agent to interactively learn to search for relevant objects in a scene through biasing the bottom-up attention and the object recognition part in order to find its state and to choose physical actions accordingly. In particular, we use the U-TREE algorithm [26] to dynamically discretize the visual state space when perceptual aliasing occurs. Aliasing means that two perceptions demanding different actions are classified under the same state. That way an object-based binary tree is generated which is used for controlling top-down object-based visual attention. Our model is inspired by the abstract findings from neuroscience and psychology.

In Section 2 of this paper, related researches are reviewed. Our proposed approach is explained in Section 3. Experiments and results are shown in Section 4 and finally, Section 5 summarizes and concludes the paper.

2. Related researches

In this section, we review studies which are directly related to ours, especially those which have considered learning aspects of visual attention in concert with decision making. First we review some existing hypotheses and viewpoints on visual attention mainly derived from behavioral studies and then bring some successful approaches from AI for learning attention control.

An important evidence from biology reported in [13], states that attention could be learned by past experience. In a behavioral task, human subjects were supposed to answer a question about a quality of a specific visual item in a synthetic visual search scene. Subjects had lower reaction times when the quality of the object stayed the same during successive trials. This study shows that subjects developed a memory during the task. A modeling work trying to explain such behavioral data is done in [27]. They have proposed an optimization framework to minimize an objective function which is a sum over the reaction time in each state weighted by the probability of that state to occur. Then using a Bayesian Belief Network (BBN), they solved that minimization problem. These results encourage using a learning approach for attention control in AI.

Some RL studies have previously been proposed for modeling top-down visual attention control in humans. Since eye movements have high correlation with overt visual attention, these studies have tried to explain eye movement data. In [28], RL is used for modeling the behavior of an expert reader by predicting where eyes should look and how long they should stay there for achieving best comprehension from the text. Another model of human eye movements is proposed in [29] that directly ties eye movements to the ongoing demands of behavior.

RL has also been used for deriving visual attention policies for mobile robots. In [30], a 3-step architecture is proposed for an object recognition task. First, it extracts potential focuses of interest (FOI) according to an information theoretic saliency measure. Then it generates some weak object hypotheses by matching the information at the FOIs with codebooks. The final step is done using Q-learning with the goal of finding the best perceptual action according to the search task. In [31], two approaches are proposed in a robotic platform with neck, eyes and arms for attention control. The first approach is a simple feedforward method which uses back-propagation learning algorithm while the second one uses reinforcement learning and a finite state machine for state space representation. In [32], another robotic platform containing articulated stereo-head with 4 degrees of freedom is presented which

can select the region of interest, perform attention shift with saccadic movements, build a map out of the environment and update it according to current observation.

An approach for learning gaze control for a mobile robot is proposed in [33], which proposes a model of selective attention for visual search tasks. Their model is implemented using a fixed pan-tilt-zoom camera in a visually cluttered lab environment, which samples the environment at discrete time steps. The agent has to decide where to fixate next merely based on visual information, in order to reach the region where a target object is most likely to be found. The model consists of two interacting modules. In the first module, RL learns a policy on a set of regions in the room for reaching the target object. By selecting an appropriate gaze direction at each step, this module provides top-down control in the selection of the next fixation point. The second module performs "within fixation" processing, based exclusively on visual information. An interesting point with this work is that it has incorporated learning where to look in a visual search task. Another advantage of this work is its implementation on a working robotic agent.

A bottom-up visual attention model known as saliency-based model, which is an extension and implementation of an earlier model of visual attention introduced by Koch and Ullman [5], is proposed in [21]. This model is based on the saliency concept and mimics the overall structure of the early visual system for detecting the locations which convey more visual signals and are different from their surroundings. Simplicity and little computation are the two main advantages of this model. It has been continuously updated and is the basis of the newer models. It has also been used to explain behavioral data on simple synthetic and static search arrays to dynamic natural stimuli like movies and games [34]. In this study we use this model to build our top-down attention control system upon it. In [23], a task-based model of visual attention control is proposed based on the saliency model to add top-down capabilities to it. Given a task definition in the form of keywords, this model first determines and stores the task-relevant entities in working memory using prior knowledge stored in a long-term memory. It then attempts to detect the most relevant entity by biasing its visual attention system with the entity's learned low-level features. It attends to the most salient location in the scene, and attempts to recognize the attended object through hierarchical matching against object representations stored in the long-term memory. It updates its working memory with the task-relevance of the recognized entity and updates a topographic task relevance map with the location and relevance of the recognized entity. Instead of a predefined definition in the form of keywords or a sentence for a task, we would like to learn manipulations over the basic saliency map by rewards and punishments that the agent receives. This approach is more general because it allows the agent to interact with the environment and to find its own way of achieving an unknown goal in a need-based manner.

In [35], Walther et al. have proposed an approach for learning and recognition of multiple objects in cluttered scenes using the saliency-based model of visual attention. They have modified the saliency model to find spatial salient regions which are more likely to contain an object. That way, the modified model could consider the extent of the objects at the focus of attention. Then they have used the SIFT features [36] for recognition of multiple objects in the scene. In their approach attention selects the extent of an object and then these objects are incrementally learned and added to a repository of learned objects. By comparing the performance of David Lowe's recognition algorithm, with and without attention, they have shown that their approach can enable one shot learning of multiple objects from complex scenes. In this regard, our research is an extension of this work in terms of detection and learning of multiple objects in a scene. The main difference is that selection of such objects is task specific and is learned interactively. In another work [37], the same authors have combined the saliency model with the standard model of object recognition. Their main idea was to consider the shape of the attended object in order to shape the area of attention. Their main claim has been based on the experimental findings that attention could be tied to objects, object parts or groups of objects. It weakens the previous belief that recognition before attention makes no sense [11,38]. To model this effect they have introduced a model for attending to salient proto objects. "Proto-objects" or "pre-attentive objects" are a step above the mere localized features. In fact proto-objects possess some but not all of the characteristics of objects. They have utilized the model of object recognition in cortex [39] to use the proto-objects to recognize multiple objects in the scene.

In [49], Jodogne et al. have presented a framework known as RLVC (reinforcement learning of visual classes) for learning mappings from images to actions by interacting with the environment. RLVC consists of two interleaved learning processes: (1) an RL unit which learns image to action mappings and (2) a binary image classifier which incrementally learns to distinguish visual classes when perceptual aliasing occurs. The classifier acts like an attention tree by checking whether a specific SIFT feature is present in the image or not. RLVC is the extension of a previous seminal work known as U-TREE algorithm [26] to visual domain. The main idea behind both approaches is that state-space is incrementally discretized whenever aliasing occurs. The main drawback with the RLVC is its exhaustive search over entire image for computing SIFT features which contradicts the philosophy of existence of visual attention. In our method, like RLVC, an action-based binary decision tree is constructed for checking the existence of some objects in a scene in an attentional framework.

3. Proposed model

An agent working in an environment receives information momentarily through its visual sensor. It should determine what to look for in every scene. For this we use RL to teach the agent to look sequentially for the most task-relevant entities in the visual scene. Our model consists of three layers: early visual processing, higher visual processing, and finally decision making and learning layer as illustrated in Fig. 1. In order to keep the model simple but functional, only important interconnections among components are shown. This model is an extension of our earlier model to interactive environments when an agent has to perform physical actions in response to perceived visual information [40].

An example scenario of the model is as follows; see Fig. 1. In the early vision layer, captured scene of the environment through the agents' visual sensor undergoes a biased bottom-up saliency detection operation and a focus of attention area is selected. For reducing the computational complexity, higher visual processes (like object recognition) are targeted only at the focus of attention (FOA). After the attended object is recognized (i.e. is either present or not in the scene), then the agent moves in its binary tree in the decision making and learning layer. This is done repetitively until it reaches a leaf which determines its state. The best motor action is this state is performed. Outcome of this action over the world is evaluated by a critic and a reinforcement signal is fed back to the agent to update its internal representations (attention tree) and action selection strategy in a quasi-static manner. Following subsections discuss each layer of the model in detail.

3.1. Early visual processing layer

The biased bottom-up attention component in this layer first determines the saliency of all spatial locations, then selects the most salient region (FOA), and finally sends it to the higher level vision for further processing. The biases are selected by the top-down

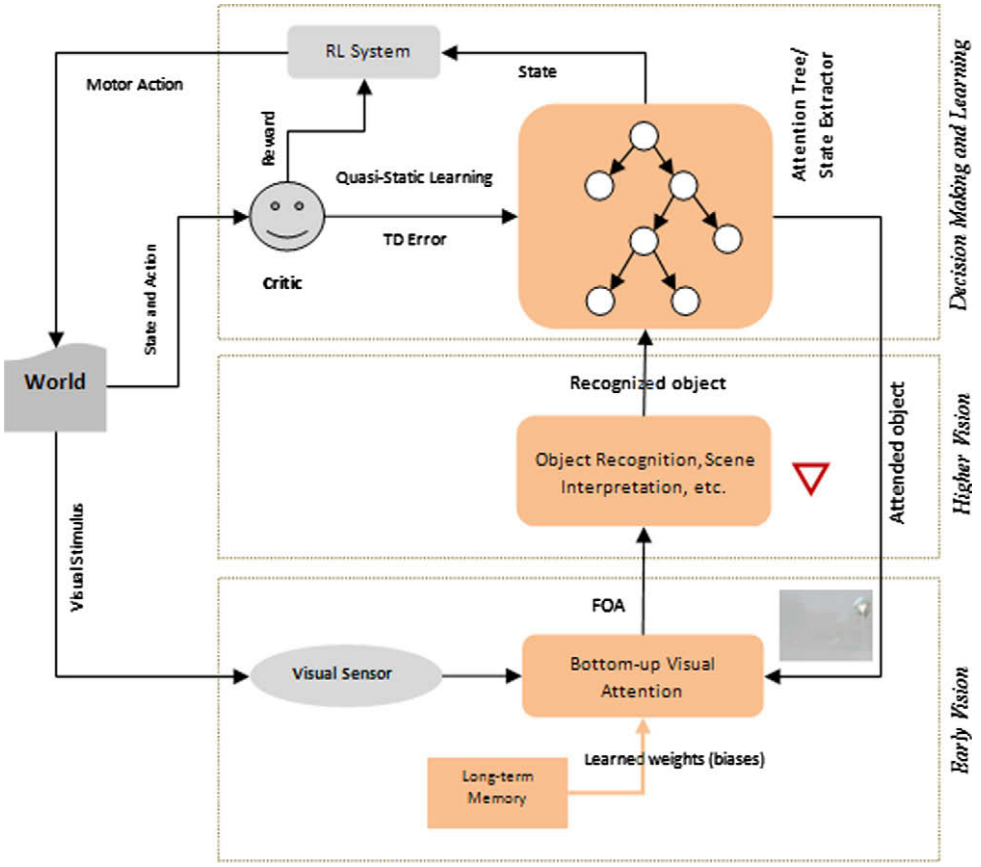

Fig. 1. Proposed model for learning task-driven object-based visual attention control.

attention part through attempting to find a desired object in the scene. As a result, the biases should be learned for each object and are task independent. Therefore, to keep the online learning resources for the task dependent parts, an offline method for learning the biases is adopted and a long-term memory is used by the bottom-up attention component to store the learned biases for online use. The early vision unit is the artificial counterpart of the V1 and V2 cortical areas in the visual cortex (what/where pathways).

Saliency model in presence of no bias selects a spatial location in a feed forward manner. In [24], the basic model is biased toward a specific object by maximizing the SNR ratio of the object to the whole image. In [48], we modified the basic model and then used a global optimization technique to learn biasing weights of the saliency model offline. Instead of only finding the appropriate weights, we have also incorporated processing costs of the feature channels to force the optimization process to choose the feature vectors with high detection rate and low cost. Below we briefly describe our biasing approach for fast object detection in a scene.

An overall sketch of our modified saliency model is presented in Fig. 2. The input image to the system is decomposed into 3 feature channels: Intensity (I), Color (C) and Orientation (O). Color channels are calculated as follows. If r, g and b are the red, green and blue dimensions in RGB color space, then $I = (r+g+b)/3$, $R = r-(g+b)/2$, $G = g-(r+b)/2$, $B = b-(r+g)/2$, and $Y = r+g-2(|r-g|+b)$ (negative values are set to zero). Local orientations (O_θ) are obtained by applying Gabor filters to the images in the intensity pyramid I. These operations are shown in (1). P_s is the feature map at scale s. P could be intensity (I), Red (R), Green (G), Blue (B), Yellow (Y) or orientation (O). $o_{\theta,s}$ is the orientation channel at orientation θ and scale s:

$$F_{I,s} = SI(I_s), F_{RG,s} = SI(R_s - G_s), F_{BY,s} = SI(B_s - Y_s), F_{\theta,s} = SI(o_{\theta,s}) \quad (1)$$

In (2), $F_{I,s}$, $F_{RG,s}$, $F_{BY,s}$ and $F_{\theta,s}$ are the intensity, red/green, yellow/blue and orientation channels in scale s, respectively. SI is the surround inhabitation operation; see [48] for details. These feature maps are summed over scales and the sums are normalized again:

$$F_l = N(\Sigma_s (s\omega)_s \cdot F_{l,s}) \quad \text{with} \quad l \in L_I \cup L_C \cup L_O \quad \text{and} \quad L_I = \{I\}, L_C = \{RG, BY\}, L_O = \{0^\circ, 45^\circ, 90^\circ, 135^\circ\} \quad (2)$$

where $(s\omega)_s$ is the weight of scale s. $N(.)$ is an iterative, nonlinear normalization operator, simulating local competitions between neighboring salient locations [42]. In each feature channel, feature dimensions contribute to the conspicuity maps by weighting and normalizing once again (3):

$$C_p = N\left(\sum_{l \in L_p} (d\omega)_p \cdot F_l\right), p \in \{I, C, O\} \quad (3)$$

Variable $(d\omega)_p$ in (3) determines weight of a dimension within feature channel p. All conspicuity maps are weighted and combined at this stage into a final saliency map:

$$SM = \sum_k (c\omega)_k \cdot C_k, k \in \{I, C, O\} \quad (4)$$

$(c\omega)_k$ in (4) weights the influences of feature channels in the final saliency map. The locations in the saliency map compete for the highest saliency value by means of a *Winner-Take-All* (WTA) network of integrate and fire neurons [5]. The winning location of this process is attended to and the saliency map is inhibited at this location. Continuing WTA competition, the next most salient location is sequentially attended to form a scanpath of successive overt attentions.

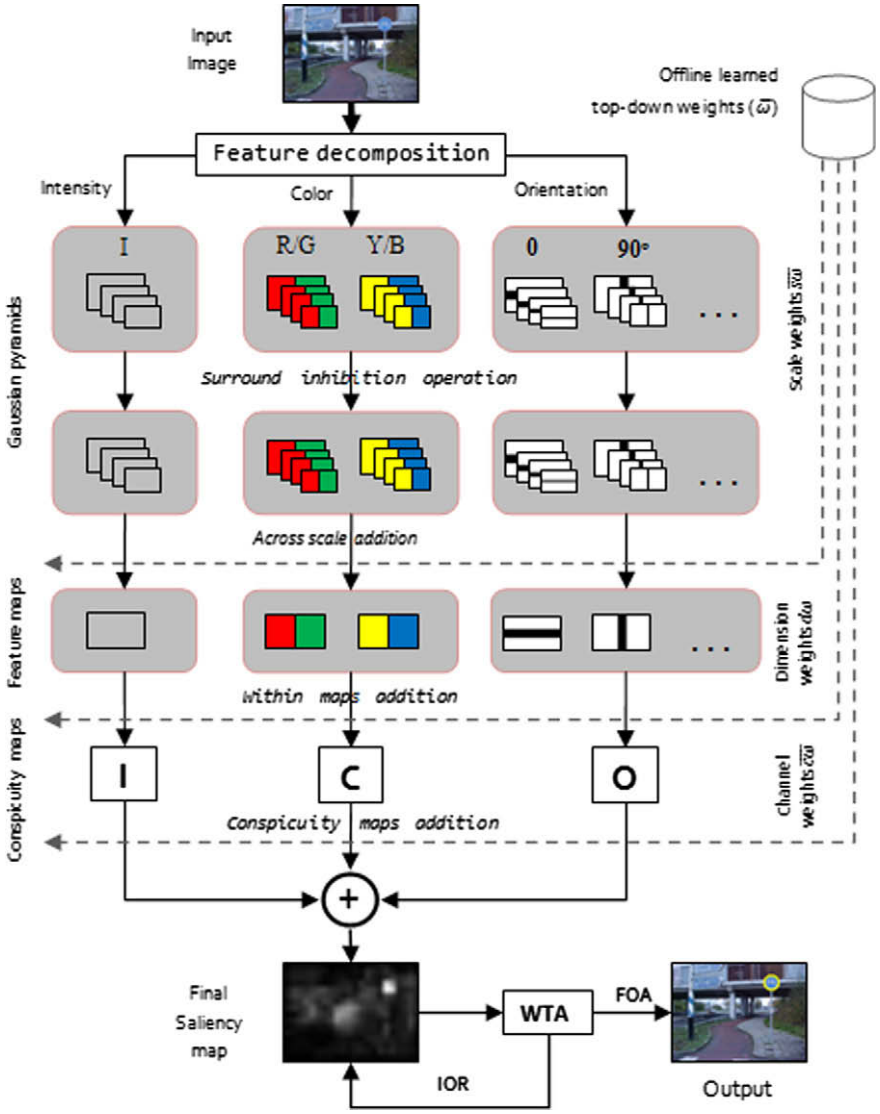

Fig. 2. Proposed biased bottom-up saliency model.

Weight vector ($\bar{\omega}$) determining weights of the model has to be learned to make an object of interest salient:

$$\begin{aligned}&\bar{\omega} = (\overline{c\omega}, \overline{d\omega}, \overline{s\omega}), \quad |\bar{\omega}| = 16, \quad |\overline{c\omega}| = 3, \quad |\overline{d\omega}| = 7, \quad |\overline{s\omega}| = 6\\ &d\omega = (\overline{d\omega I}, \overline{d\omega C}, \overline{d\omega O}), \quad |\overline{d\omega I}| = 1, \quad |\overline{d\omega C}| = 2, \quad |\overline{d\omega O}| = 4\end{aligned} \tag{5}$$

In (5), $\overline{c\omega}$, $\overline{d\omega}$, and $\overline{s\omega}$ are weight vectors for feature channels, dimensions within channels and scales, respectively. $\overline{d\omega I}$, $\overline{d\omega C}$, and $\overline{d\omega O}$ are weight vectors for intensity, color (red/green and yellow/blue) and orientation (0°, 45°, 90° and 135°) dimensions. Our aim is to find a weight vector which maximizes the object detection rate over a set of M training images in which the location of the target object is already tagged. For this purpose, we follow a data-driven approach. First, the optimal weight vector is sought to satisfy an objective over a training image dataset and is then evaluated over a separate set of test images. Assume that training set T contains M images with an object of interest tagged in them as:

$$T = \{(Im_1, t_1), (Im_2, t_2), \ldots, (Im_M, t_M)\} \tag{6}$$

Then the fitness of a weight vector is defined as:

$$F(\bar{\omega}) = \frac{1}{M}\left(\sum_{i=1}^{M} \text{norm}(\text{Saliency}(Im_i, \bar{\omega}) - t_i)\right) \tag{7}$$

In (7), norm (.) is the Euclidean distance between two points in an image. Saliency is the function which takes as input an image and a weight vector and returns the most salient location. t_i is the location of target object in the ith image.

Note that, a lower fitness value for the above function means that it has better performance. For minimizing the fitness function, an algorithm known as comprehensive learning particle swarm optimization (CLPSO) is used [43]. CLPSO is simple, easy to implement and has been applied to a wide range of optimization problems. Refer to [48] for the details of the implementation.

3.2. Higher visual processing layer

This layer acts upon the information extracted by the early visual processing layer. In fact, the most salient part of the scene, highlighted by top-down biasing is processed here. Operations in this layer are more sophisticated and give the agent a more cognitive understanding of the image/scene; useful for learning and decision making in the next layer. Example operations could be ob-

ject recognition, scene classification, background subtraction, etc. Processes in this unit are dependent on the environment and the tasks of the agent. This layer corresponds to cortical areas at the end of visual ventral stream like V4, IT and PFC.

Here we use the standard model of object recognition in cortex for recognizing objects at the focus of attention. Standard model of object recognition (Hierarchical Model and X (*Hmax*)) is a hierarchical model which closely follows the operations and findings from the early and late visual system. For more detailed information on *Hmax*, the interested reader should refer to [44–46]. It will be used for object recognition in visual navigation tasks in Section 4. Note that as in the saliency model, *Hmax* is also trained offline and is used online when performing a task.

The *Hmax* is used in this way; first, C2 features of an object tagged in a train set of images are extracted and then a binary SVM classifier is trained with these features and is then evaluated over a separate set of test images. When operating online, C2 features of attended location is derived and passed to the corresponding classifier to check if attended object is the same as expected object or not. Detailed setup and results of object recognition are shown in Section 4.2.

3.3. Decision making and learning layer

The core of our model is the decision making and learning layer where visual attentions and representations are learned. In addition, this layer controls both top-down object-based attention and motor actions. Top-down attention is equivalent to overt attention which in humans means that attention is relocated to different objects with eye movements. The motor actions are executed by the body of the agent and affect the world. State extractor unit (attention tree) in this layer, derives the state of the agent based on the cognitive information it receives from the higher vision unit. The learning approach is an extension of the U-TREE algorithm [26] to visual domain.

Based on the derived state, the agent must choose to either do a motor action or perform another perceptual action which here means looking for another object in the scene to further clarify its state. For example in Fig. 3a it is possible to discriminate two perceptions by checking whether object O_2 exists in the scene or not but in Fig. 3b, one further object checking is necessary to find the state. Looking for another object involves biasing the early vision by using the associated pre-learned weights (biases) of the saliency model and running the object recognition part accordingly, see Fig. 1. The agent continues traversing the attention tree until it finally reaches a leaf node which determines its state in the *Q*-table. The agent employs its *Q*-table for selecting a motor action. An evaluation of the motor action over the world is fed back to the agent by the critic in form of a reinforcement signal. This signal is used by the agent for updating its *Q*-table and developing its attention tree.

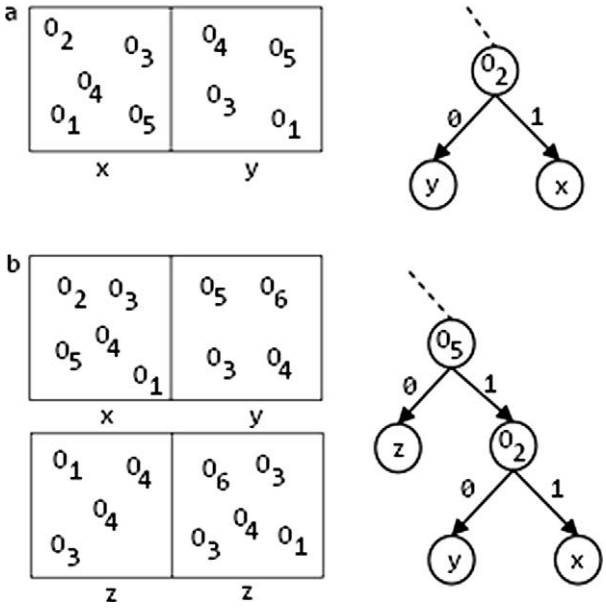

Fig. 3. Resolving aliasing by subsequent object checking: (a) one object checking is enough to discriminate two scenes which need different actions. (b) Two subsequent object checking for action-based discrimination of several scenes.

Since there are two interleaved parameters in the model, attention tree and the *Q*-table, we adopted a quasi-static approach for learning the attention tree and image (state) to action mappings (*Q*-table). Each time reinforcement signal from the critic is used for either modifying the tree or updating the *Q*-table. This layer corresponds to decision making areas in the brain like LIP and PFC cortices.

3.3.1. Learning attention tree

An efficient method to implement attention and state space construction is by means of *tree* data structures. Such structures are interesting because they allow learning representations and attention control at the same time. Visual discretization is achieved via expanding the attention tree whenever perceptual aliasing occurs. Such refinement is performed to increase the cumulative discounted reward of the agent in each time step t:

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}, \text{where } \gamma \in [0,1] \tag{8}$$

where $0 < \gamma < 1$ is the discount factor [19].

Each internal node of the tree checks whether a specific object exists in the scene or not. The generated tree for attention control is a binary tree, since an object checked in an internal node of the tree either exists or not in the scene. When an image is presented to the agent, attention tree is sequentially traversed from the root until a leaf node is reached.

Attention tree is incrementally built in a quasi-static manner in two phases (iterations): (1) *Tree-fixed* (*RL-update*) phase and (2) *RL-fixed* (*Tree-update*) phase. In each phase of the algorithm, the feedback of the critic (a scalar reward) is used to alternatively update the policy or refine the leaves with aliasing.

The algorithm starts with a single node (state) in the *Tree-fixed* phase and then moves to the *RL-fixed* phase and so on, see Table 1. It means all the images are initially mapped to that single node. It is obvious that such a single state is not enough in many cases and therefore aliasing occurs. Then, the algorithm breaks that node into two leaves in the *RL-fixed* phase based on some gathered experiences beneath it. In each *Tree-fixed* phase, RL algorithm is executed for a number of episodes according to the learned *Q*-table from the previous phase by following an ε-greedy action selection strategy [19]. In this phase, the tree is hold fixed. It means, the agent perceives its state using the fixed attention tree, performs motor action a_t, receives reward r_{t+1} and enters state s_{t+1} where $(s_t = [o_1, o_2, \ldots, o_n])$, $o_i \in \{0,1\}$, and o_i is 1 if object o_i is detected by the higher vision in the scene. The derived quadruples $s_t, a_t, r_{t+1}, s_{t+1}$ are used for updating the *Q*-table according to *Q*-learning formula [50]:

$$Q(s_t,a_t) = (1-\alpha)Q(s_t,a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q(s_{t+1},a)\right) \tag{9}$$

where $0 < \alpha < 1$ is the learning rate.

State discretization occurs in the *RL-fixed* phase. In this phase, gathered experiences by the agent are used to refine leaves of the attention tree with aliasing. An important point to be considered here is that the agent only accesses the environment through its visual sensor (e.g. its CCD camera). Therefore, in order to

Table 1
Algorithm for learning top-down object-based attention control.

```
main ()
tree = Create a tree with a single node                    //all input images are mapped to this single state at the beginning
Repeat
  Tree-fixed phase                                         // in this phase only Q-table is updated
    for i = 1 to maxEpisodes
      I_t = take an image
      s_t = traverseTree (tree, I_t)
       [s_{t+1} r_{t+1} a_t] = performAction(s_t)
      Q-table = updatePolicy(Q-table, s_t)
    end                                                    // end for i
  RL-fixed phase                                           // in this phase only tree is modified
    for i = 1 to maxEpisodes
      I_t = take an image
      s_t = traverseTree(tree, I_t)                        // s_t is a leaf node
       [s_{t+1} r_{t+1} a_t] = performAction(s_t)          // based on an action selection policy choose action a_t, go to state s_{t+1} and get reward r_{t+1}
      Δ_t = calcDelta (s_t, s_{t+1}, r_{t+1}, a_t)         // Δ_t is the TD error according to Eq. (10)
      mem = gatherMem(a_t, I_t, Δ_t)                       // this item is saved for state (node) s_t
    end                                                    // end for i
  for j = 1 to size(tree.leaves)                           // expand the nodes with aliasing
  if size(mem) > memTreshold
    if checkAliasing(s_t)
      tree = modifyTree(tree, s_t)
    end                                                    // end for j
  pruneTree(tree)
Until (no more aliasing) or (maximum iterations is reached)   // repeat
```

determine its state in any position, the agent should capture a scene in that position and then traverse its attention tree from the root node down to a leaf.

3.3.2. Measuring aliasing

After each *RL-fixed*(*Tree-update*) phase, attention tree is refined by expanding the states (leaves) with perceptual aliasing (Table 2). In order to estimate aliasing, a number of patterns should be accumulated under a leaf node (*gatherMem*( ) function in Table 1). To have a better estimation for aliasing, we only refined those leaves with the memory size greater than a threshold value (*memThreshold*). Memory is updated with adding the last episode to it. An experience under a node with state s_t is $([o_1, o_2, \ldots, o_n], a_t, \Delta t), o_i \in \{0, 1\}$, where o_i is 1 if object o_i exists in the scene. In order to know if object o_j is present in the scene or not agent applies the bias vector of the jth object and then uses the corresponding SVM classifier for that object. This way agent could find which objects are in the captured scene. As in the *Tree-fixed* phase, an image is captured, attention tree is traversed in order to find the perpetual state, appropriate action is performed and a reward is received. A prominent measure of perceptual aliasing in a state (leaf node) is the TD error (also known as Bellman residual) and is derived from the Q-learning formula as shown in Eq. (10):

$$\begin{aligned} Q(s_t, a_t) &= \alpha\Big(r_{t+1} + \gamma \max_a Q(s_{t+1,a}) - Q(s_t, a_t)\Big) + Q(s_t, a_t) \\ &= \alpha\Delta_t + Q(s_t, a_t) \end{aligned} \tag{10}$$

Table 2
Function for checking aliasing.

```
checkAliasing(s_t)                                  // A is the set of all motor actions
  for action a ∈ A
    mem(a) = all memory items with action a
   under s_t
    var(a) = calcVariance(mem(a))                   // of mem(a)
    if var (a) > aliasingThreshold
      return true;
  end                                               // end for action a
  return false;
```

where Δ_t is the *TD-error* of state s_t with respect to action a_t. In order to detect aliasing, all patterns under a node are clustered according to their physical actions and then if any of these clusters has a variance in Δ_ts greater than a threshold (*aliasingThreshold*), then that node has aliasing at least with respect to one action. When the RL algorithm converges, then the Q-values do not change any more and Δ_ts vanish. It means that the final tree is achieved.

3.3.3. Tree refinement

In order to refine the attention tree, each aliased state is expanded into two leaves. Tree modification (refinement) is done by selecting the object which mostly reduces the variance in Δ_t of patterns in the memory of a leaf according to Eq. (11). In [47], splitting measures for U-TREE are compared and is shown that variance is a more effective measure in terms of generating tress with smaller number of states and less computation time than other measures:

$$\begin{aligned} o^* &= argmin_o\left(var\{y\} - \left(\frac{|L_+|}{L} var\{y_a|+\} + \frac{|L_-|}{L} var\{y_a|-\}\right)\right) \\ &= argmax_o\left(\frac{|L_+|}{L} var\{y_a|+\} + \frac{|L_-|}{L} var\{y_a|-\}\right) \end{aligned} \tag{11}$$

where y is the set of all memory items under a leaf, y_a is the set of all memory items with action a, $y_a| + (y_a|-)$ is the set of memory items with (without) object o and action a. Sizes of these two sets are $|L_+|$ and $|L_-|$, respectively. o^* is the selected object.

Maximization is done over all pairs of objects and actions. An object which minimizes the variance also has to cluster the images under the node into two populations with significantly different distributions for the best separation. Here we used t-test for comparing these two distributions. If the power of the t-test is below 0.05, then two distributions are considered to be significantly different. When expanding a node, an object is selected which has not been already used in the path from this node to the root. When a leaf is created (deleted) a corresponding state is added to (deleted from) the Q-table and the Q-values are uniformly initialized. Table 3 shows tree refinement function.

Table 3
Function for tree refinement.

```
modifyTree(tree, s_t)
  for action a ∈ A
    mem(a) = all memory items with
   action a under s_t
    for object o ∈ O                    // O is the set of all
                                        possible objects
                                        the agent might observe in
                                        a scene
       choose the object which reduces
   the variance the most according to
   Eq. (11) and also partitions the memory
   into two populations with non-equal
   distributions (using t-test)
    end                                 // for o
  end                                   // for a
```

3.3.4. Tree pruning

Proposed algorithm constructs an attention tree in a greedy manner which usually leads to overfitting. Therefore solutions should be designed to overcome overfitting by either periodic tree restructuring or pruning. Two heuristics are introduced in the following.

Consider two types of nodes: (1) nodes having leaves with the same best actions learned for them; (2) nodes with at least one child with no memory. First case happens when next *RL-update* phase assigns same best actions to leaves with the same parent and second case happens when next *Tree-update* phase does not explore part of the state space represented by one leaf. Leaves of the nodes in the first category are removed and their action is assigned to their parent node. A node in the second category is removed and its child with memory is substituted with it. These two heuristics are recursively done from bottom to top of the tree until no node satisfies one of these conditions.

4. Experimental results

In this section, we evaluate performance of each part of the model individually and then put them all together in the overall model. Performance of the proposed model is analyzed over visual navigation tasks in which the agent has to learn which object to attend to find its state and then which physical action to do at that state. First, we train the object detection and recognition parts offline (in first and second layers) based on the objects that the agent will observe when working online.

4.1. Biasing results

In this section results of biasing saliency model in the early visual processing layer for object detection in natural scenes are shown. We used 3 traffic signs (bike, crossing and pedestrian) and two objects (coke and triangle). Number of images for bike, crossing, pedestrian, triangle and coke were 70, 45, 55, 69 and 42, respectively. Sizes of images were 360×270 pixels. Fig. 4 illustrates sample signs and objects in natural scenes.

CLPSO was trained over 10 random images for each object and then the best weight vector was tested over the remaining images of that object. Results are reported over five runs with random train images. Derived weight vectors for detection of bike and crossing signs after CLPSO convergence are shown in Fig. 5. For bike, color (yellow/blue) and orientation (45° and 135°) channels have the highest weights. Middle scales (s_1 to s_3) are more informative for detection of this object. For crossing, again color channel has the highest weight. For this sign, red/green channel is more important. Orientation dimensions (0°, 45° and 135°) which appear in shape of the crossing sign have higher weights. Since almost in all images of this sign, triangle is toward up, these orientations are stable features and have got higher weights than other orientation dimensions.

Table 4 shows the average values of detection rates using fitness function in (7). An object was considered detected if a salient point was generated in a vicinity of 30 pixels around its center. For detection of an object rather than the most salient point, 2 other locations generated by WTA were also considered. It can be seen from Table 4 that biasing saliency model results in higher object detection rates compared with basic saliency model over all objects. Since object detection is not perfect, errors in detection have to be considered for online learning. For online experiments we used a subset of data where object detection could be done perfect and then analyzed behavior of the model in presence of detection errors. For more comprehensive results of biasing the saliency model refer to [48].

4.2. Object recognition results

In this section, object recognition component in higher visual processing layer is evaluated. Object recognition is applied to the object at the focus of attention to make sure that recognized object is the attended one. This knowledge helps the agent to move in its attention tree and to find its state.

Since locations of target objects in the training images used for biasing the saliency model in early vision layer were already known we used this information for training the SVM classifiers. First, C2 features from 10 random training images for each object were extracted in a 50 × 50 window around the center of object in each scene. This was done for all objects. Then for each object, C2 features from training images were tagged as positive samples and the C2 feature vectors from all other objects from their training images were considered negative. A binary SVM classifier [51] was trained using all training samples of the target class (positive patterns) and training samples from other classes (negative patterns). The trained classifier was later tested over remaining test patterns of the same class (with positive label) and negative test patterns of all other classes. Location of the object is just known in offline training. Window is put around the center of the object and the whole image patch at that area is used for training the classifier. Process of training a SVM classifier for the bike sign is shown in Fig. 6.

Classification using the C2 features resulted in 91.28% (±2.8%), 93% (±2.75%), 87% (±3.2%), 83% (±4.2%) and 94.6% (±1.4%) recogni-

Fig. 4. Sample objects in natural scenes. From left to right bike, crossing, pedestrian, coke and triangle. Target is shown by the yellow circle.

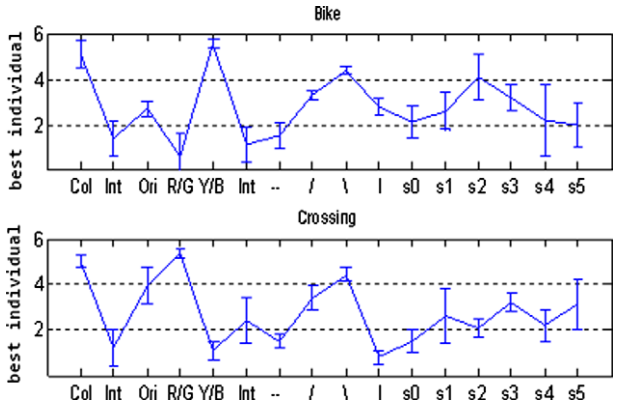

Fig. 5. Learned weights after CLPSO convergence over first two traffic signs averaged over five runs. s_0 to s_5 are scales in the image pyramid.

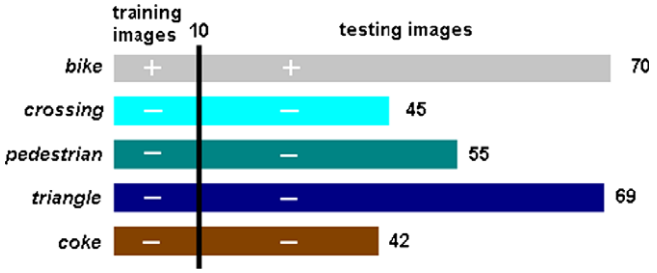

Fig. 6. Binary SVM training for the bike class. For each object 10 images are selected in random for training. Here, bike SVM classifier is trained with positive samples from bike class and negative samples from training images of other classes and then is tested with positive and negative test images.

tion rates for bike, crossing, pedestrian, triangle and coke objects, respectively. Reported results are averaged over 5 runs with random training patterns for each object.

4.3. Visual navigation task

In this experiment, the agent is supposed to learn how to navigate safely to the goal (G) in a simulated driving environment. The agent uses its offline learned knowledge – biasing weights and the trained object recognition part – interactively. Map of the route, consisting of 4 positions $(1, 2, 3, G)$, is shown in Fig. 7. The agent has no access to its (x, y) position and only receives 360×270 *RGB* color images of natural scenes in each situation. At each state, the agent has three possible motor actions: *Forward* (F), *Turn Left* (L) and *Turn Right* (R). It can attend to one of several objects in the images each time. For checking the existence of an object, when moving in its attention tree, the agent applies the learned biasing signals for that object to the observed scene (long-term memory component in the model). Then the most salient region of this biased saliency map is transferred to the corresponding trained binary SVM classifier (associated with the attended object), and based on a positive or negative answer from this classifier moves in the attention tree until it reaches a leaf node determining its state.

Since there are 3 positions (minus Goal) and 4 sides per each one, there are $3 \times 4 = 12$ states in the environment. The agent in each of the 12 states captures a scene containing either one or more objects. In each episode of RL, the agent is placed randomly in a position of the map. States of the environment are numbered in this way: $(x - 1) \times 4 + y$, where x is the position number (in the central circle) and y is the head direction of the robot (in the coordinate system at the bottom-right of Fig. 7). $y = 1$, 2, 3 and 4 for North, West, South and East directions, respectively. For instance, state number of north side of position 2 is 5 and for its east, it is 8. Note that the best actions in the image are with respect to the head direction of the agent. For example, when head of the agent is toward east in the map, south lies in its right side. Note that, the optimal policy is not unique in this map. It means, more than one action might be the best for a state. They are shown with red characters $\{F, R, L\}$ besides each state in Fig. 7.

An example assignment of sets of objects – pedestrian (A), crossing (B), bike (C) and triangle (D) – to states is shown in Fig. 7. For each subset, we captured 5 natural scenes containing those signs. These scenes could also contain other signs but important signs for the agent in this experiment are only the four signs shown in Fig. 7. Examples of the natural scenes used in this experiment are shown in Fig. 8. Actually, scenes were selected in such a way to give the agent a deterministic behavior.

In this experiment, we assumed that agent is capable of detecting and recognizing objects in the scenes. It means that when the agent applies the learned biases for a specific object to the bottom-up attention model, it can correctly detect and recognize that object in the scene at the first saccade. The reason is that uncertainties due to perceptual errors are not yet studied in U-Tree algorithm and it is not subject of our study.

We solve this problem is two cases. In the first case mapping of motor actions to states are already known and only states of the environment should be discovered. This case is a supervised version of the problem when associated actions are known by the supervisor. This assumption is relaxed in the second case and both attentions and motor actions are learned for safe navigation.

4.3.1. Experiment I: motor actions known and fixed

Navigation task is a coupled task since it demands learning both state representations and motor actions. In this section, we aim to solve the case in which motor actions for states are already known. Therefore it only remains to learn the states and their associations with motor actions. Then we relax this assumption and use our model for learning both representations and their associated motor actions in Section 4.3.2.

The agent starts with a single *Null* state and all captured images are mapped to that state. For a randomly captured image, the agent must choose between doing a motor action and attending to another object to clarify its state (to increase the chance of correct classification of this image). In the *Null* state, the agent is only allowed to do attentions (and not motor actions), since it is clear that more than one state is needed for representing the environment.

Table 4
Comparison of detection performances of biased saliency model and the basic saliency model over natural objects and traffic signs with max fixations equal to 3. Results are averaged over 5 runs with random training and test sets. Numbers in parentheses are standard deviations.

		Target Object				
		Bike	Crossing	Pedestrian	Coke	Triangle
Biased Saliency Model	Train	92.3(2.1)	96.7(1.2)	98.2(1)	95.2(1.4)	92.5(2.3)
	Test	90.2(2)	93.8(0.9)	94.2(1.1)	92.2(2)	91(1.6)
Basic Saliency model	Test	81.8(0.6)	78.2(1.4)	83.3(1)	80.9(0.4)	76.5(0.8)

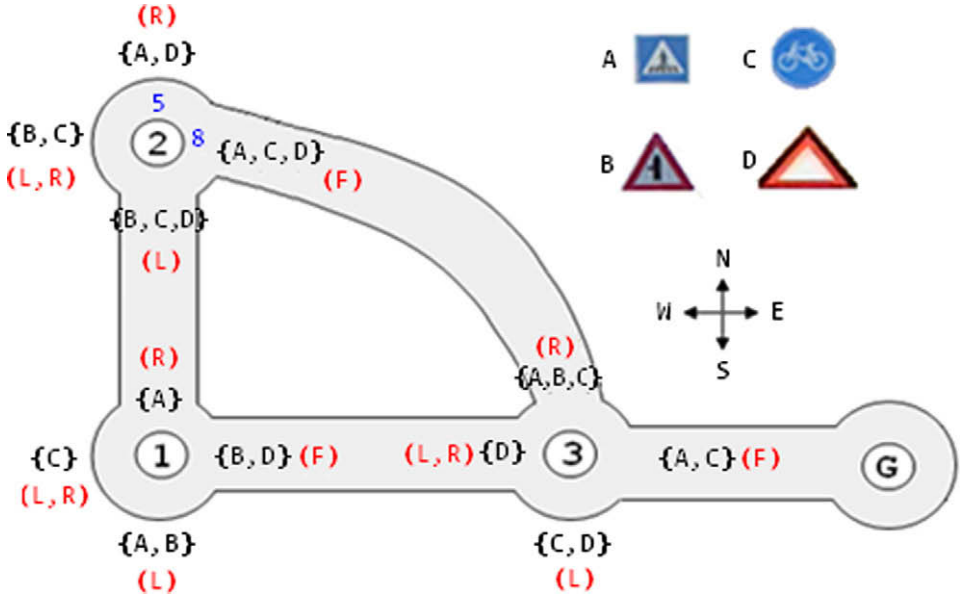

Fig. 7. Simple navigation map. Goal state is shown with capital letter G. Best actions are shown in parentheses and objects in the natural scenes associated to states in brackets. The agent can go forward, turn left or right and can only attend to one object in the scene each time. It should reach the goal state as fast as possible.

This causes the algorithm to generate other states. In each state, the agent performs either an attention or a motor action using ε-greedy action selection policy. If predicted motor action by the agent for this state is optimal (best), it receives a big reward of +100 and otherwise it gets −100 punishments. If it attends to an object, it receives a punishment which is the cost associated with looking to that object. Then, the agent updates the Q-table entry for this state and moves to the next state. If the next state does not exist in the Q-table, then it is created and initialized uniformly. This process is repeated for some episodes (*maxEpisodes*). Flowchart of Fig. 9 illustrates this process.

Cumulative average rewards and average costs for this experiment are shown in Fig. 10. Results are compared with the situation when the agent has the full observation (full attention). In this case, the agent checks the existence of (attends at the same time to) all objects in the scene in order to find its state. A '1' means that the attended object exists in the scene and '0' means it does not. Since the response for attending to an object is either 1 or 0, state space (with 4 objects) in full attention case has 2^4 states.

As shown in Fig. 10, both attended and full-observation cases, converged to the same average reward in iteration 2000 (bottom panel). The reason why both curves are below +100 is because of the punishments the agent receives when performing the task. Top panel of Fig. 10 shows the average cost of both approaches. When the agent attends to all objects, it always has the cost of $(-10 = -4 - 3 - 2 - 1)$. So, its average cost is always −10. When it attends to one object at a time, it has a higher average cost at first, but as long as it learns to do the task (and to attend), its average cost reduces and finally reaches a value above −10. The agent was capable to navigate correctly after learning.

The final behavior of the agent after learning is explained next. When observing an image, it starts from the *Null* state and then follows its learned policy in the attention tree until a motor action is selected. If in a state an object should be attended, the agent first applies learned top-down gains for that object to the image, then a square area around the salient part (50×50 pixels) is sent to the SVM classifier corresponding to that object. Then next state is determined and so on. If the state does not exist it is created.

4.3.2. Experiment II: motor actions unknown

In this section, association of physical actions to scenes is not known in advance. Therefore, the agent has to discover both its representations (internal state space) and the best image-to–action mappings. The agent starts in the *Tree-fixed* phase with a random position and a random head direction in the map. It then receives an image randomly from 5 images associated with each state. To find its state, the agent traverses its attention tree down to a leaf node and then it follows an ε-greedy strategy for action selection. Each episode is terminated when the agent reaches the goal state or goes off the road (hits the wall). Rewards and punishments are defined as:

$$R\begin{cases} +100: & \text{reaching the goal state} \\ -10: & \text{going forward and hitting a blocked way} \\ -5: & \text{turning and observing a blocked way in front} \\ 0: & \text{ordinary forward or turn} \end{cases} \tag{12}$$

Average reward is the sum of the rewards that the agent receives during an episode normalized by the length of that episode. Average tree depth is the average depth of all leaf nodes in the attention tree. Fig. 11 shows the average reward of the agent, statistics of the generated tree like number of nodes and leaves and its average depth after running the algorithm in Table 1. It also shows error rate in the policy of the agent. In this figure each iteration consists of 35 episodes and in each episode the agent starts from a random location and moves until it reaches the goal or hits a wall. Fig. 11 (top-panel) shows the smoothed cumulative reward in a window of 20 episodes.

Final depth of the generated tree is 3.54, which means that the agent could do the task perfectly by attending to 3.54 objects in average. This value is below 4 objects in full attention case and shows near 12% improvement in reducing the processing costs. As the attention tree grows, measured by average tree depth, the correct policy rate increases. As Fig. 11 (middle-panel) shows average depth of the tree increases during learning. Fig. 11 (bottom-panel) shows the percentage of correct policy rate. After 12 iterations, it converges to 100% rate that means the agent has learned the optimal image to action mapping and there are no further aliasing

Fig. 8. Sample natural scenes with traffic signs. Five images for each state were randomly selected. Co-occurrence of some signs together determines the state of the agent.

in the tree. Final generated tree is shown in Fig. 12. Eleven leaves (states) were generated for handling the task which is smaller than 12 states of the environment. In fact, since some states have the same best actions, they were clustered under the same leaf nodes (state with plus sign in Fig. 12).

It could be verified from the tree in Fig. 12 that the algorithm generates inefficient trees. In its general form, it creates some unnecessary leaf nodes. For example, the node labeled with $*$ in this figure has two children with the same best actions. Clearly, such nodes have anomaly and could be merged and be replaced with their parent node. Some nodes also do not absorb any experiences indicating that they are not necessary. To generate compact trees, generated trees are pruned in some occasions, for instance after d iterations (here $d = 3$).

Generated tree in a new run with pruning is shown in Fig. 13. Algorithm in this case succeeded to generate a more compact tree by clustering more states with the same best actions under the same leaf nodes. Final generated tree again achieved 100% correct policy rate with 6 states and average depth of 2.66 which are smaller than 3.54 and 11, average depth and number of states of the unpruned tree in Fig. 12.

We also applied our method to a more complex navigation environment. In this task, the agent moves between 11 positions of the map in Fig. 14. Neither motor actions nor visual representations are known in advance. The state space is of size $11 \times 4 = 44$. Actions and reinforcement signals are as in the previous section. The agent has to connect an input image to the appropriate reaction without explicitly knowing its geographical location.

Five scenes are associated with each state and there is no error in detection and recognition. Objects in the scenes (4 traffic signs plus the coke object) are in random locations in 2D space and are not bound to specific spatial locations. We put the triangle and

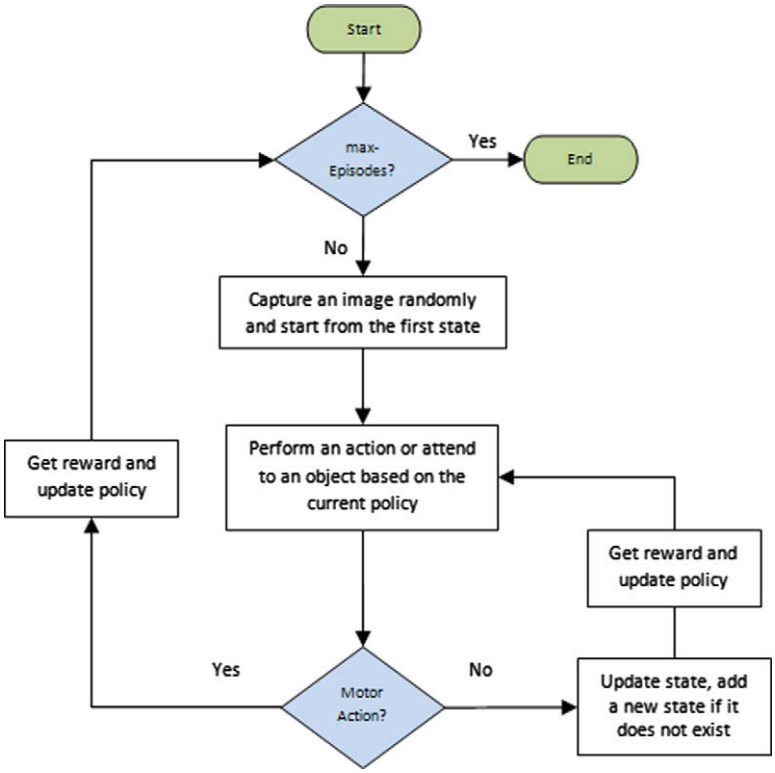

Fig. 9. Flowchart of the algorithm for learning attention control when motor actions are known in advance. State of the agent is determined by the objects it observes. The agent finally learns to do a motor action or to attend to another object in each state when it is presented with an image.

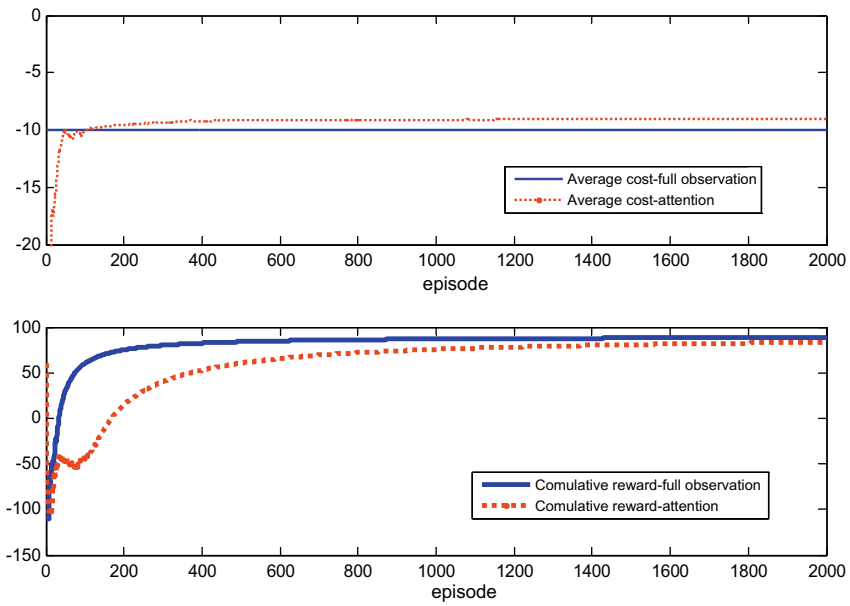

Fig. 10. (Top) Average cost, (Bottom) Cumulative average reward during episodes. RL parameters: *maxEpisodes* = 2000, $\alpha = 0.7$, $\gamma = 0.9$, $\varepsilon = 0.9$. Costs were defined as [−4 −3 −2 −1] for objects [A B C D].

coke objects in different natural scenes and then photographed the image to make different assignments.

A random assignment of the objects to the states is shown in Table 5. Note that, the agent observes some same scenes in different states. Since there are 44 states in the map, and 31 combinations are possible with 5 objects (minus empty scene), therefore aliasing occurs in some states (the agent observes the same scene in different states with the same best actions). To avoid such aliasings, no same scenes were assigned to states with different optimal actions.

Generated tree for this experiment without pruning and with deterministic observation (without uncertainty) has 23 leaves.

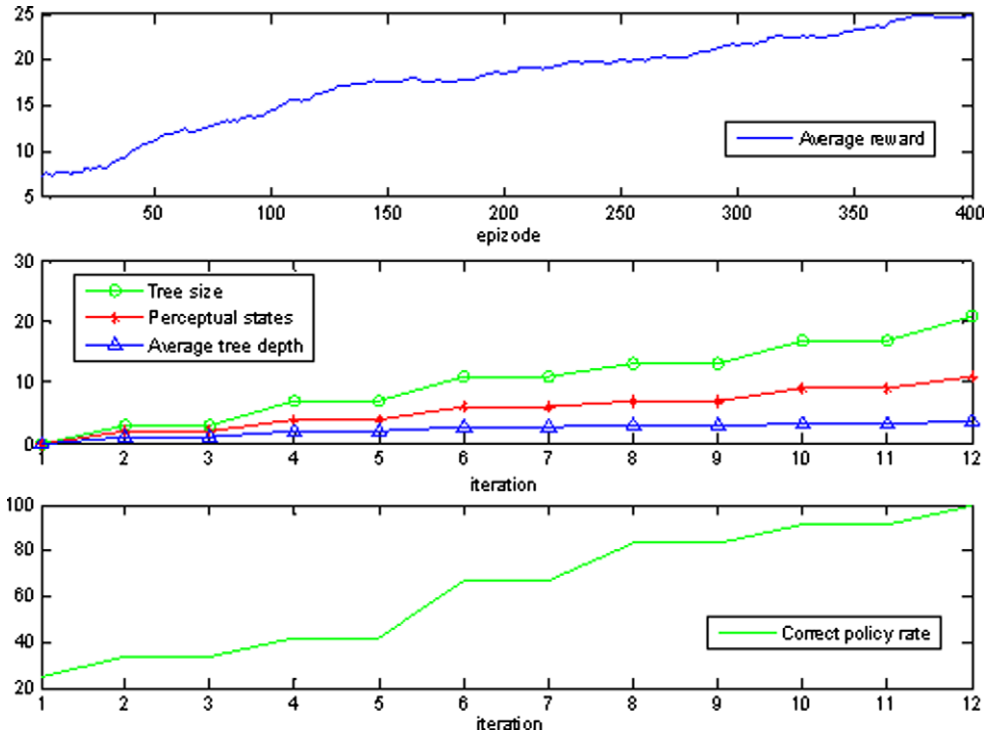

Fig. 11. (Top) Smoothed average reward in a window of 20 episodes for *RL-Update* iterations. (Middle) Number of nodes and leaves of the learned attention tree and average tree depth. (Bottom) Percentage of the correct policy rate. Parameters: *maxEpisodes* = 35, $\varepsilon = 0.8$, *memThreshold* = 20, $\alpha = 0.9$, $\gamma = 0.9$, *aliasingThreshold* = 11 (for the map of Fig. 7, without pruning).

Algorithm converged to 100% correct policy rate after 55 iterations (phases). Final tree has the average depth of 4.2 which is smaller than 5 objects (full attention). So, the agent attends to 4.2 objects in average while having the same performance.

Fig. 15 shows the generated tree after pruning. Algorithm generated 7 states with average depth of 3. Therefore pruning could extensively help optimizing the trees.

4.4. Experiment III: uncertainty analysis

An important aspect of decision making in real-world situations is dealing with uncertainty in sensors. For instance, a major source of uncertainty for a robotic agent is its sensors, which are often noisy and have only a limited view of the environment. Since both saliency model and the *Hmax* have uncertainties, this problem also

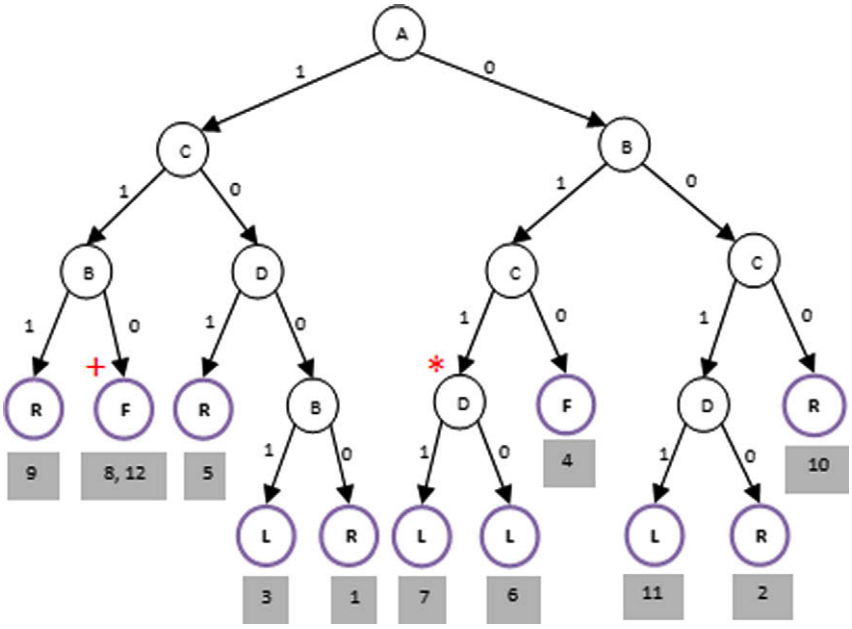

Fig. 12. Learned attention tree for the map of Fig. 7 without pruning. Algorithm managed to generate a tree with 11 states. Letters inside the blue circles (leaves) are the learned motor actions. Characters inside the internal nodes are names of objects. Numbers below leaves are state numbers. Numbers on the edges determine whether object has been seen or not. This tree resulted in 100% correct policy but is not efficient since the node marked with $*$ has two children with the same best actions. Different states with the same best actions are clustered under the same leaves.

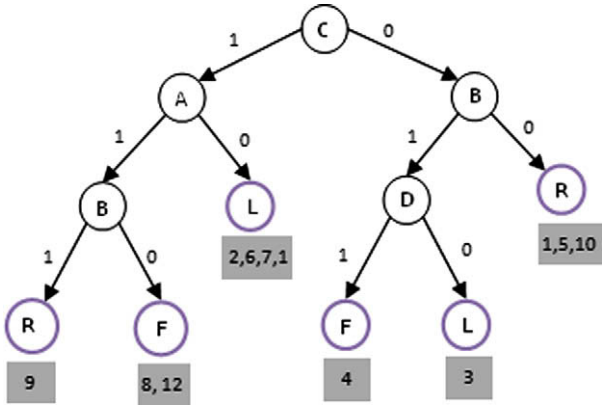

Fig. 13. Learned attention tree for the map in Fig. 7 with pruning. Twelve states were clustered into 6 leaves. Algorithm succeeded to achieve the 100% correct policy in this case. Parameters: *maxEpisodes* = 200, $\varepsilon = 0.8$, *memThreshold* = 100, $\alpha = 0.9$, $\gamma = 0.9$, *aliasingThreshold* = 10.

applies to our model. In this section, we analyze how uncertainty in the perception of the agent affects its behavior over the map of Fig. 7. Each observation of the agent is incorrect by probability P_u. For example, $P_u = 0.03$ means that in 3 percent of observations the agent is not sure that an object is really present in the scene or not. When the agent traverses its attention tree and has to attend to an object, it gets an incorrect result with probability P_u. When observations of the agent are noisy, then the agent develops a probabilistic action selection strategy. Since no formulation is involved, the only way for the agent to discover such probabilities is by maximizing its reward. Results of re-running experiment in Section 4.3.2 with uncertainty in object detection is shown in Fig. 16.

In this experiment, tree generation was stopped after 12 iterations whether algorithm converged or not for comparing the uncertainty levels. That is why correct policy rate is not 100% in the bottom panel. As Fig. 16 shows (top and bottom panels), low magnitude of noise does not degrade the behavior of the agent a lot (for $P_u = 0$ and $P_u = 0.05$). However when the noise uncertainty is increased to 0.1 or 0.4, algorithm does not converge to a near optimal policy. Increasing the uncertainty resulted in gaining less average reward and more faulty policies. Thus, the agent could compensate low magnitude of uncertainty in the saliency model and *Hmax* by optimizing its action selection strategy.

5. Summary and conclusions

In this research, we proposed and implemented an overall biologically inspired model for top-down object-based visual attention control. In our model, we considered how task demands, actions and the bottom-up cues influence attention. Our results support the idea that the nature of the bottom-up attention is low-level mechanisms, while top-down attention is more like a control or a decision making problem. To solve such problems optimization approaches have been followed in engineering literature. Rather than scanning the image from top-left to bottom-right, to detect an object in the scene, or using global representations (which usually need many computations), our model just looks at a small number of spatial locations. The information about what and where to look comes from the top-down knowledge learned by RL during learning a task.

Our approach is in accordance with the important biological evidence that brain has adopted a need-based approach for visual representations. This means that representations and other cognitive capabilities of the agent are altered or refined in order to optimize its behavior and interactions with the world it lives in. A minimal solution (representation) is enough and no further detail (over-completeness) is needed. This mechanism is well captured by our model for deriving visual representations and top-down task-based attentions. Actually representations and attentions are optimized in our model by maximizing cumulative reward of the agent (behavior-based approach). For gaining maximum reward those states with aliasing are expanded into further states which then lead to sufficiently finer representations.

Major next step is to extend the work to continuous interactive visual environments where the agent continuously receives visual information from the environment and has to perform continuous

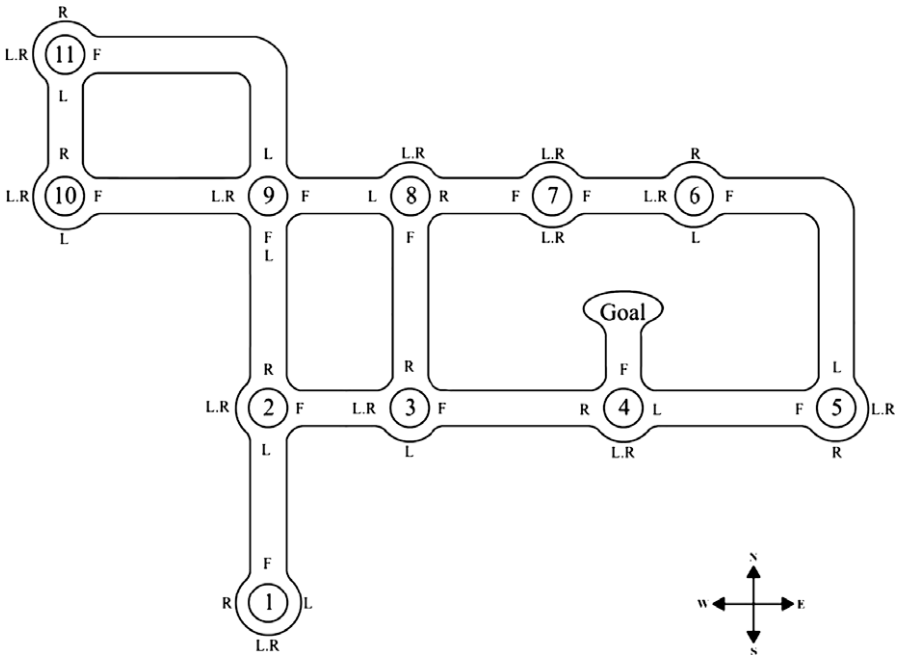

Fig. 14. Complex navigation map. A subset of 5 objects is randomly were present in natural scenes. Best actions are shown besides each state. In some states two actions are optimal.

Table 5
A random assignment of objects to states of the map in Fig. 14.

1	{B}	9	{A, B, D}	17	{A, C, E}	25	{A, B, C, E}	33	{A, B, D}	41	{A, B, C, D}
2	{A, B}	10	{A, B, C, E}	18	{B, C, D, E}	26	{D, E}	34	{A, D, E}	42	{A, C, D}
3	{A, C, D}	11	{A, C, E}	19	{D}	27	{A, D, E}	35	{D, E}	43	{A, C, E}
4	{A, C}	12	{B, C, D}	20	{A, D, E}	28	{B, D, E}	36	{B, D, E}	44	{B}
5	{A, B, C, D}	13	{B, D}	21	{A, B, D}	29	{A, D, E}	37	{A, B, C, D}		
6	{A}	14	{A, E}	22	{A}	30	{A, D}	38	{A, C, D}		
7	{A, C}	15	{A, D, E}	23	{A, C, E}	31	{B, C, D, E}	39	{A, C, E}		
8	{B, D, E}	16	{A, D}	24	{B, D}	32	{A, B, D}	40	{B, C, D}		

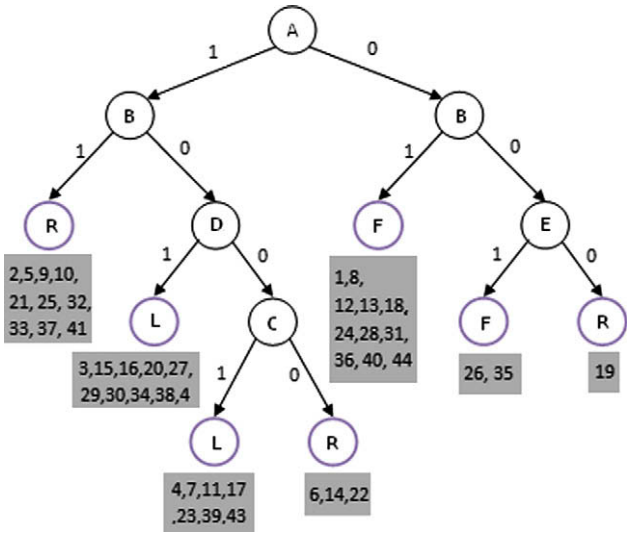

Fig. 15. Learned attention tree for the map of Fig. 14Fig. 14 after pruning. Forty-four states were clustered into 7 leaves. 100% correct policy was achieved.

and complex motor actions. Since the biasing approach is capable of detecting various objects, it could be applied for tasks where several objects are important for the agent. Relationships among objects could be considered in the higher vision layer in order to increase the set of perceptual states.

Integration of bottom-up and top-down attentional mechanisms has not been studied and shown in previous computational models which leaves place for future works. For example, a computer operator has learned how and where to move his eyes when working with a computer program from his past experience. This shows the task-driven component of visual attention. When suddenly he closes a screen which needs something to be saved, suddenly a pop-up message box appears and alarms for saving. Our top-down model could be tried to explain the top-down component of this sample task using eye movement data. However, how the bottom-up part should be integrated in the model needs further investigation.

For state determination agent learned to sequentially check the existence of objects in the order of decreasing perceptual aliasing. In some cases it is not enough to know which objects are present in the scene to find the state. It is possible that the same set of objects in a scene convey a different meaning depending to their configuration. We have anticipated this in the higher vision layer of the model which should be accommodated for different tasks and visual behaviors. In this regard it is also interesting to add the *gist* concept to the model. Gist is the capability of the human to classify a visual scene in a very short presentation (about 80 ms) [41]. This could be used for bypassing the successive object checking and jumping to the lower levels of the tree to increase classification speed.

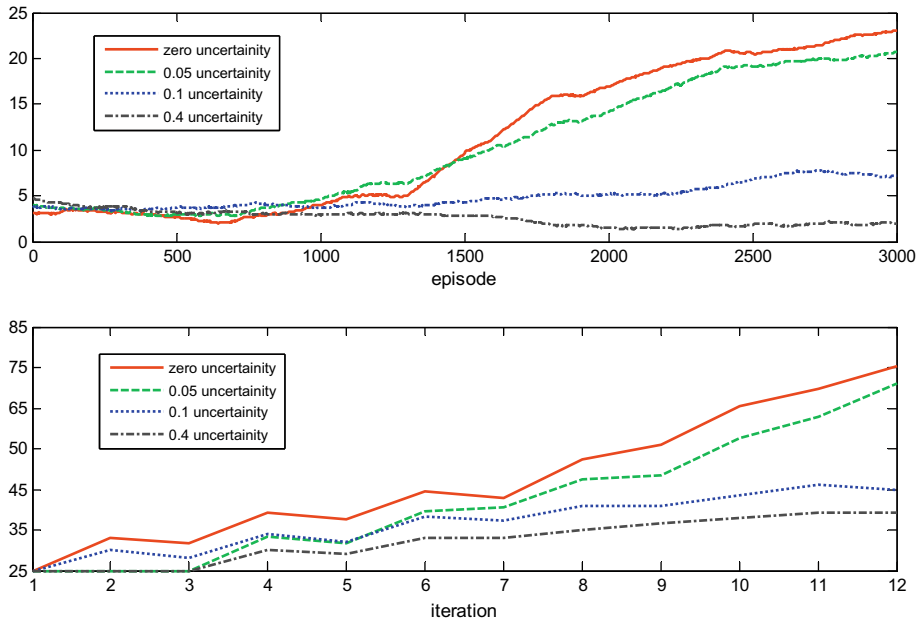

Fig. 16. (Top) Cumulative average reward of the agent for different noise levels. (Bottom) Cumulative percentage of correct policy during learning (for the map of Fig. 7). Results are averaged over 7 runs. Analysis was done using pruning and the same parameters as in Fig. 13.

In this paper, we only considered the perceptual aliasings mediated by the insufficient observations. If the agent could observe all the necessary data, therefore no aliasing occurs. Some aliasings are due to lack of knowledge about previous actions and observations. For example when two same scenes, associated with two states with different best actions, are presented to the agent, then it could not discover its optimal policy because of the contradiction in decision making. Such problems are remedied in RL by equipping the agent with a short-term memory that keeps track of the previous actions, attentions and observations to satisfy the Markov property. This idea could be used to extend our model for performing tasks with this type of aliasing.

References

[1] H.E. Egeth, S. Yantis, Visual attention: control, representation, and time course, Annual Review of Psychology 48 (1997) 269–297.
[2] C.E. Connor, H.E. Egeth, S. Yantis, Visual attention: bottom-up versus top-down, Current Biology 14 (2004) 850–852.
[3] M. Corbetta, G.L. Shulman, Control of goal-directed and stimulus-driven attention in the brain, Nature Reviews 3 (2002) 201–215.
[4] M.M. Chun, J.M. Wolfe, Visual attention, in: E.B. Goldstein (Ed.), Blackwell's Handbook of Perception, vol. 9, Blackwell, Oxford, UK, 2001, pp. 272–310.
[5] C. Koch, S. Ullman, Shifts in selective visual attention: towards the underlying neural circuitry, Human Neurobiology 4 (1985) 219–227.
[6] Z. Li, A saliency map in primary visual cortex, Trends in Cognitive Sciences 6 (2002) 9–16.
[7] S. Kastner, L.G. Ungerleider, The neural basis of biased competition in human visual cortex, Neuropsychologia 39 (2001) 1263–1276.
[8] M.I. Posner, Orienting of attention, Quarterly Journal of Experimental Psychology 32 (1980) 3–25.
[9] J.H. Maunsell, S. Treue, Feature-based attention in visual cortex, Trends in Neurosciences 29 (2006) 317–322 (TINS special issue: The Neural Substrates of Cognition, 2006).
[10] N. Kanwisher, J. Driver, Objects, attributes, and visual attention: which, what, and where, Current Directions in Psychological Science 1 (1992) 26–31.
[11] J.H. Duncan, Selective attention and the organization of visual information, Journal of Experimental Psychology: General 113 (1984) 501–517.
[12] A.L. Yarbus, Eye movements during perception of complex objects, in: L.A. Riggs (Ed.), Eye Movements and Vision, Plenum Press, New York, 1967, pp. 171–196 (Chapter VII).
[13] V. Maljkovic, K. Nakayama, Priming of pop-out: I. Role of features, Memory & Cognition 22 (1994) 657–672.
[14] E. Gibson, E. Spelke, The development of perception, Handbook of Child Psychology vol. iii: Cognitive Development, Wiley, 1983 (Chapter 1).
[15] W.D. Gray (Ed.), Integrated Models of Cognitive Systems, Oxford University Press, New York, 2007.
[16] R. Pfeifer, J.C. Bongard, How the Body Shapes the Way We Think a New View of Intelligence, MIT Press, 2006.
[17] A. Clark, Where brain, body, and world collide, Journal of Cognitive Systems Research 1 (1999) 5–17.
[18] J. Triesch, D.H. Ballard, M.M. Hayhoe, B.T. Sullivan, What you see is what you need, Journal of Vision 3 (2003) 86–94.
[19] R.S. Sutton, A.G. Barto, Reinforcement Learning, The MIT Press, Cambridge, MA, 1998.
[20] B. Seymour, J.P. O'Doherty, P. Dayan, K. Koltzenburg, A.K. Jones, R.J. Dolan, K.J. Friston, R.S. Frackowiak, Temporal difference models describe higher order learning in humans, Nature 429 (2004) 664–667, doi:10.1038/nature02636.
[21] L. Itti, C. Koch, E. Niebur, A model of saliency-based visual attention for rapid scene analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 1254–1259.
[22] L. Itti, C. Koch, Computational modeling of visual attention, Nature Reviews Neuroscience 2 (2001) 195–203.
[23] V. Navalpakkam, L. Itti, Modeling the influence of task on attention, Vision Research 45 (2005) 205–231.
[24] V. Navalpakkam, L. Itti, An integrated model of top-down and bottom-up attention for optimizing detection speed, Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference 2 (2006) 2049–2056.
[25] A. Torralba, Modeling global scene factors in attention, Journal of Optical Society of America A 20 (2003) 1407–1418 (special issue on Bayesian and Statistical Approaches to Vision).
[26] A.K. McCallum, Reinforcement learning with selective perception and hidden state, Doctoral dissertation, Department of Computer Science, University of Rochester, 1995.
[27] M.C. Mozer, M. Shettel, S.P. Vecera, Top-down of visual attention: a rational account, in: Y. Weiss, B. Schoelkopf, J. Platt (Eds.), Neural Information Processing Systems, vol. 18, 2005, pp. 923–930.
[28] E.D. Reichle, P.A. Laurent, Using reinforcement learning to understand the emergence of intelligent eye-movement behavior during reading, Psychological Review Copyright 2006 by the American Psychological Association 113 (2006) 390–408.
[29] N. Sprague, D.H. Ballard, Al. Robinson, Modeling embodied visual behaviors, ACM Transactions on Applied Perception 4 (2) (2007).
[30] L. Paletta, G. Fritz, C. Seifert, Cascaded sequential attention for object recognition with informative local descriptors and Q-learning of grouping strategies, Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).
[31] L.M.G. Gonic, G.A. Giraldi, A.A.F. Oliveira, P.A. Grupen, Learning policies for attentional control, IEEE International Symposium on Computational Intelligence in Robotics and Automation (1999) 294–299.
[32] L.M.G. Gonic, A. Antonio, A.A.F. Oliveira, P.A. Grupen, A framework for attention and object categorization using a stereo head robot, Proceedings on the XII Brazilian Symposium on Computer Graphics and Image Processing (1999) 143–152.
[33] S. Minut, S. Mahadevan, A reinforcement learning model of selective visual attention, Fifth International Conference on Autonomous Agents, Montreal, 2001.
[34] R.J. Peters, L. Itti, Applying computational tools to predict gaze direction in interactive visual environments, ACM Transactions on Applied Perception 5 (2) (2008) (Article 8).
[35] D. Walther, U. Rutishauser, C. Koch, P. Perona, Selective visual attention enables learning and recognition of multiple objects in cluttered scenes, Computer Vision and Image Understanding 100 (2005) 41–63.
[36] D. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[37] D. Walther, C. Koch, Modeling attention to salient proto-objects, Neural Networks 19 (2006) 1395–1407.
[38] R. Egly, J. Driver, R.D. Rafal, Shifting visual attention between objects and locations: evidence from normal and parietal lesion subjects, Journal of Experimental Psychology General 123 (2) (1994) 161–177.
[39] M. Riesenhuber, T. Poggio, Hierarchical models of object recognition in cortex, Nature Neuroscience 2 (11) (1999) 1019–1025.
[40] A. Borji, M.N. Ahmadabadi, B.N. Araabi, Interactive learning of top-down attention control and motor actions, Workshop on From motor to interaction learning in robots, IROS 2008.
[41] A. Oliva, A. Torralba, Building the gist of a scene: the role of global image features in recognition, Progress in Brain Research: Visual Perception 155 (2006) 23–36.
[42] L. Itti, C. Koch, Feature combination strategies for saliency-based visual attention systems, Journal of Electronic Imaging 10 (1) (2001) 161–169.
[43] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 9 (2006) 3.
[44] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Object recognition with cortex-like mechanisms, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (3) (2007) 411–426.
[45] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, T. Poggio, A Theory of Object Recognition: Computations and Circuits in the Feedforward Path of the Ventral Stream in Primate Visual Cortex, AI Memo 2005-036/CBCL Memo 259, Massachusetts Institute of Technology, Cambridge, MA, 2005.
[46] A. Borji, M. Hamidi, F. Mahmoudi, Robust handwritten character recognition with features inspired by visual ventral stream, Neural Processing Letters 8 (2) (2008) 97–111.
[47] M. Asadpour, M.N. Ahmadabadi, R. Siegwart, Reduction of learning time for robots using automatic state abstraction, in: H.I. Christensen (Ed.), Proceedings of the First European Symposium on Robotics, vol. 22, Springer Tracts in Advanced Robotics, Palermo, Italy, Springer-Verlag, 2006, pp. 79–92.
[48] A. Borji, M.N. Ahmadabadi, B.N. Araabi, Cost-sensitive learning of top-down modulations for attention control, Machine Vision and Applications, doi:10.1007/s00138-009-0192-0.
[49] S. Jodogne, J.H. Piater, Closed-loop learning of visual control policies, Journal of Artificial Intelligence Research 28 (2007) 349–391.
[50] C. Watkins, P. Dayan, Q-learning, Machine Learning 8 (3) (1995) 279–292.
[51] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.
[52] C. Fermüller, Y. Aloimonos, Vision and action, Image and Vision Computing 13 (10) (1995) 725–744.